Tk

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/046,200	01/16/2002	John O. Lamping	108759	3057

27074 7590 04/20/2006

OLIFF & BERRIDGE, PLC.
P.O. BOX 19928
ALEXANDRIA, VA 22320

EXAMINER	
MITCHELL, JASON D	
ART UNIT	PAPER NUMBER
2193	

DATE MAILED: 04/20/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	**Application No.** 10/046,200	**Applicant(s)** LAMPING, JOHN O.	
	Examiner Jason Mitchell	**Art Unit** 2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *08 March 2006*.

2a)□ This action is **FINAL**. 2b)☒ This action is non-final.

3)□ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-9* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)□ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-9* is/are rejected.

7)□ Claim(s) _____ is/are objected to.

8)□ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)□ The specification is objected to by the Examiner.

10)□ The drawing(s) filed on _____ is/are: a)□ accepted or b)□ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)□ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)□ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)□ All b)□ Some * c)□ None of:

1.□ Certified copies of the priority documents have been received.

2.□ Certified copies of the priority documents have been received in Application No. _____.

3.□ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) □ Notice of References Cited (PTO-892)

2) □ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) □ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
Paper No(s)/Mail Date _____.

4) □ Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ .

5) □ Notice of Informal Patent Application (PTO-152)

6) □ Other: _____.

DETAILED ACTION

This action is in response to remarks filed on 3/8/06.

At applicant's request, claims 1, 7 and 8 have been amended. Claims 1-8 are pending.

Response to Arguments

Applicant's arguments on pp. 6-8 regarding the 35 USC 103 rejection of claims 1, 7 and 8 have been fully considered but they are not persuasive.

In the first full paragraph on pg. 7, Applicant states:

> However, a close examination of Biggerstaff shows that there is no teaching of aspect-oriented programming anywhere in the Biggerstaff reference.

Examiner respectfully disagrees. While Biggerstaff has labeled his techniques 'Anticipatory Optimization' he is addressing the same problem as Applicant. For example compare Applicant's paragraph 5:

> When compiled, the modular units used in object-oriented programming roughly correspond to separate blocks of code. Since these blocks of code are executed without regard to the structure of other blocks, needless redundant operations often occur. For example, an object may do an addition operation and then a multiplication operation on a vector of operands. In the object-oriented programming paradigm, this would probably be accomplished using separate methods, one for addition and one for multiplication, both involving the use of arrays. First, the addition operands would be read in from the arrays. Then, the addition method would execute and store its results in an intermediate array. Subsequently, the multiplication method would execute and read in the addition method's results. Then, it would perform the multiplication and store the results in a third array. In this simple example, there were four read or write operations to the arrays.

With Biggerstaff's col. 2, lines 11-51:

> For example, to double the value of each element of a matrix and then add the resulting values to the values of the of the elements of another matrix, ... the compiler may generate nested "for" loops to select the element at each row and each column of the C matrix, to multiply the value of the selected element by 2, and

> to store the doubled values in a temporary matrix. The compiler also generates nested "for" loops to select the element at each row and column of the temporary matrix, to add the values to the value of the corresponding element of the B matrix ... The compiler generates a third set of nested "for" loops to select the element at each row and column ... and to store the value of that element into the A matrix.

Although Biggerstaff uses matrixes instead of arrays, it is clear that Biggerstaff is concerned with the same type of 'needless redundant operations' caused by the compiler's inability to detect the repeated aspects of the calculation.

Further, Biggerstaff offers solution (col. 4, lines 1-2 'loop merging') that is at least similar to the solution proposed by Applicant (par. [0007] 'loop fusion'), and provides this solution using the steps claimed by Applicant as indicated in the rejection. Further because, Biggerstaff discloses all of the claimed limitations, except an explicit use of the words "Aspect Oriented", it is Examiners position that Biggerstaff does in fact disclose an Aspect Oriented method.

Applicant goes on to state:

> Moreover, Biggerstaff does not teach simplifying a programming element by reducing the programming element to a canonical expression

Examiner respectfully disagrees. In col. 8, lines 22-26, Biggerstaff discloses "transforming high-level, domain operators and operands into successively lower-level operators and operands". This disclosure, coupled with his disclosure that "a matrix is conceptually a composition of low-level computational constructs, such as integers" (col. 7, lines 43-44) indicates that his transformation into lower-level operators and operands (col. 8, lines 22-26) is analogous to Applicants disclosure in paragraph [0031] ("determining the value of "(1+2)*4" simplifies first to "3*4", and then to "12", which is the canonical representation of the number twelve").

Applicant goes on to state:

> Finally, Biggerstaff does not disclose or suggest associating at least one projection that gives information about the role of a mathematical expression in the simplification of the programming element

Examiner respectfully disagrees. Biggerstaff's disclosure in col. 12, lines 19-17 ('the quantifier tag ... would cause a transform ... to invoke a transformation named "_PromoteAboveLoop"') clearly discloses that his projections ('quantifier tag') provide information about the role of a mathematical expression in the simplification ('invoke ... "_PromoteAboveLoop"').

Based on the rationale given above, The rejections of independent claims 1, 8 and 9 as well as the rejections to dependent claims 2-7 are maintained.

Claim Rejections - 35 USC § 101

Applicant's amendments were sufficient to overcome the 35 USC 101 rejections of claims 1 and 3-6, which are consequently withdrawn.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 1-8 are rejected under 35 U.S.C. 103(a) as being unpatentable over US 6,745,384 to Biggerstaff (Biggerstaff) in view of "Aspect-oriented Requirements Engineering for Component-based Software Systems" by Grundy (Grundy).

Regarding Claim 1: Biggerstaff discloses a method embodied on a recordable medium for simplifying an aspect-oriented programming element that is compliable into instructions for operating a data processing device, the programming element having at least one part, comprising: simplifying the programming element by reducing the programming element to a canonical expression (col. 8, lines 22-26, 'transforming high-level, domain operators and operands into successively lower-level operators and operands'; col. 7, lines 43-44 'a matrix is conceptually a composition of low-level computational constructs, such as integers') until all of the at least one part of the programming element reaches a first simplification stage to create at least one part of a current stage simplified programming element (col. 9, lines 25-29 'Inline Functions component'); determining at least one propagator for the current stage simplified programming element, (col. 7, lines 13-16 'allows a programmer to define various transforms for transforming an IP tree ... into a tree that facilitates the implementation of various optimizations') the propagator being a form that is matched against terms of an appropriate stage (col. 6, lines 8-11 'transformations are triggered by the pattern of ... operators and operands') and posting information about projections of one or more of the terms (col. 6, lines 13-14 'add property tag adornments ... that anticipated how those expressions might be implemented'); associating at least one projection with the current stage simplified programming element using the at least one determined propagator (col. 6, lines 11-14 'These transformations may ... add property tag adornments') the at least one projection giving information about the role of a mathematical expression in the simplification of the programming element (col. 12, lines 19-17 'the quantifier tag ... would cause a transform ... to invoke a transformation named "_PromoteAboveLoop"'); simplifying the current stage simplified programming element, based at least in part on the current stage simplified programming element and the associated projections (col. 6, lines 28-35 'col. 6, lines 11-14 'adornments that anticipate how those expressions might be implemented'), until all of the at least one part of the current stage simplified programming element reaches a next stage to create a next stage simplified programming element (col. 6, lines 39-46 'Once this ... loop merging is complete, the optimized code for the loops is generated').

Biggerstaff does not explicitly disclose the propagator being described in the programming element, but does disclose the propagator is defined by a developer (col. 7, lines 13-16 'allows a programmer to define various transforms').

Grundy discloses a method of describing the propagator in the programming element (pg. 6, col. 2, par. 2 'Aspect information can be encoded in component implementations'), in an analogous art for the purpose of providing an application with information regarding possible aspects (pg. 6, col. 2, par. 2 'Components my query other components for the aspects they provide or require').

It would have been obvious to a person of ordinary skill in the art at the time of the invention to use the method taught in Grundy for describing the propagator (pg. 6, col. 2, par. 2) to allow the programmer disclosed in Biggerstaff (col. 7, lines 13-16) to describe the propagator because one of ordinary skill in the art would have been motivated to provide a method to allow the developer to define this information (Biggerstaff col. 7, lines 13-16 'allows a programmer to define various transforms').

Regarding Claim 2: The rejection of claim 1 is incorporated; further, Biggerstaff discloses compiling each stage obtained from the programming element into at least a portion of the instructions for operating the data processing device (col. 6, lines 39-46 'the optimized code for the loops is generated').

Regarding Claim 3: The rejection of claim 1 is incorporated; further Biggerstaff discloses repeating the determining, associating and current stage simplifying steps using the next stage simplified programming element as the current stage simplified programming element (col. 6, lines 39-46 'Then the composite folding phase operates on the body of the resultant loops').

Regarding Claim 4: The rejection of claim 3 is incorporated; further Biggerstaff discloses repeating the determining, associating and current stage simplifying steps until the next stage simplified programming element is a final stage of the programming element (col. 6, lines 39-46 'Then the composite folding phase operates on the body of the resultant loops to simplify, rewrite, reorganize and merge').

Regarding Claim 5: The rejection of claim 1 is incorporated; further Biggerstaff discloses using the at least one determined propagator to decorate the current stage simplified programming element with the at least one projection (col. 6, lines 11-14 'These transformations may ... add property tag adornments').

Regarding Claim 6: The rejection of claim 1 is incorporated; further Biggerstaff discloses each simplified programming element has at least one significance (col. 6, lines 8-11 'individual transformations'). Biggerstaff further discloses determining whether, for each of the at least one part of the current simplified programming element, that part of the current simplified programming element should be reduced (col. 6, lines 8-11 'transformations are triggered by the ... operators and operands at each subtree') so that the next stage simplified programming element properly denotes the at least one significance of that part of the current simplified programming element in the next stage simplified programming element (col. 6, lines 7-8 'distinct recursive walks of the abstract syntax tree').

Regarding Claim 7: Biggerstaff discloses a method for executing an aspect-oriented computation on a data processing device described as a plurality of language constructs (col. 49, lines 36-41 'the resulting loops are woven together into a series of terms'), comprising: determining at least one propagator for the computation, the propagator being a form that is matched against terms of an appropriate stage (col. 6, lines 8-11 'transformations are triggered by the pattern of ... operators and operands') and posting information about projections of one or more of the terms (col. 6, lines 13-14 'add property tag adornments ... that anticipated how those expressions might be implemented'), the propagator being included in the language constructions (col. 7, lines 13-16 'allows a programmer to define various transforms for transforming an IP tree ...

into a tree that facilitates the implementation of various optimizations'); generating a projection on the computation, the projection specifying a second computation (col. 49, lines 36-41 'the common index expressions') and giving information about the role of a mathematical expression in simplifying the language constructs (col. 12, lines 19-17 'the quantifier tag ... would cause a transform ... to invoke a transformation named "_PromoteAboveLoop"'); executing the computation until a portion of the computation using the propagator that is conditional on a result of the projection is reached (col. 49, lines 36-41 'those terms simplified via partial evaluation'); simplifying the language constructs describing the computation, by reducing the language constructs to canonical expressions (col. 8, lines 22-26, 'transforming high-level, domain operators and operands into successively lower-level operators and operands'; col. 7, lines 43-44 'a matrix is conceptually a composition of low-level computational constructs, such as integers') sufficiently to allow the second computation specified by the projection to be executed (col. 49, lines 36-41 'the common index expressions within those terms are replaced with temporary variables'); executing the second computation to obtain the result for the projection (col. 49, lines 36-41 'whose values are computed'); and continuing the execution of the computation based on the obtained result for the projection (col. 49, lines 36-41 're-computation of the index expressions').

Regarding Claim 8: Biggerstaff discloses a method for converting an aspect-oriented programming element into a plurality of woven code blocks the woven code blocks compliable into instructions for operating a data processing device, comprising:

(a) identifying at least one common process (col. 6, lines 27-28 'loop merging') in the programming element;

(b) reducing the programming element to at least one significance based on the identified at least one common process (col. 6, lines 8-11 'individual transformations are triggered'), the at least one significance comprising a canonical expression (col. 8, lines 22-26, 'transforming high-level, domain operators and operands into successively lower-level operators and operands'; col. 7, lines 43-44 'a matrix is conceptually a composition of low-level computational constructs, such as integers');

(c) incorporating the at least one significance into a first woven code block (col. 6, lines 39-42 'optimized code ... is generated');

(d) determining zero, one or more of the incorporated significances that are susceptible to updating in subsequent steps of the method (col. 8, lines 31-34 'the subtree will be flagged');

(e) invoking a propagator, based upon results of the determination (col. 6, lines 8-11 'individual transformations are triggered'), usable to perform any desired updates on the determined susceptible significances of the first woven code block (col. 6, lines 11-13 'These transformations may ... transform one abstraction'); the propagator being a form that is matched against terms of an appropriate stage (col. 6, lines 8-11 'transformations are triggered by the pattern of ... operators and operands') and posting information about projections of one or more of the terms (col. 6, lines 13-14 'add property tag adornments ... that anticipated how those expressions might be implemented'), the projections giving information about the role of a mathematical expression in reducing the programming element (col. 12, lines 19-17 'the quantifier tag ... would cause a transform ... to invoke a transformation named "_PromoteAboveLoop"'); repeating steps (a)-(e) at least once to create a subsequent woven code block based on the immediately previously created woven code block (col. 6, lines 39-46 'code for the loops is generated ... Then the composite folding phase operates on the ... loops'), further comprising:

(f) communicating with the propagator of at least one previously created woven code block to determine if any significances of that at least one previously created woven code block are common to the subsequent woven code block (col. 8, lines 31-34 'the subtree will be flagged); and

(g) updating any significances in at least one of the subsequent woven code block and at least one previously created woven code block (col. 8, lines 31-34 'scheduled for further transformation') that are common to the subsequent woven code block and that at least one previously created woven code block (col. 6, lines 39-46 'code for the loops is generated ... Then the composite folding phase operates on the ... loops').

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Jason Mitchell whose telephone number is (571) 272-3728. The examiner can normally be reached on Monday-Thursday and alternate Fridays 7:30-5:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (571) 272-3719. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Jason Mitchell
4/6/06

KAKALI CHAKI
SUPERVIS... ...MINER
TECHNOLOG...